Lecture 13: Self-Organising Maps

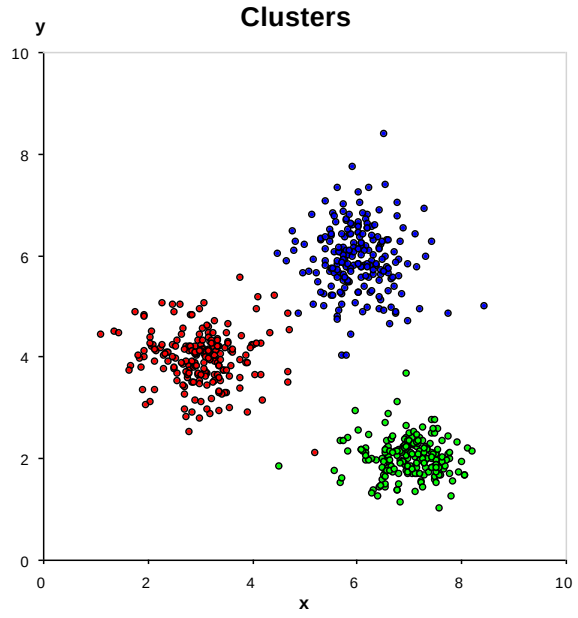
Dr. Roman V Belavkin

BIS3226

Contents

1 Data Visualisation and Topology Preserving Maps

Limitations of k-Means Clustering



- How many clusters should we be looking for?

$$k = ?$$

- How large are the clusters?
- What is the shape of the clusters?
- How are they related to each other?

Remark 1. *If the data 'lives' in a m-dimensional space with $m > 3$, then there is no way we can visualise this data directly. Thus, if we want to visualise the dataset, we need to reduce the number of dimensions to 3 or even 2.*

Topology Preserving Maps

- A topology is a system of subsets $O \subset X$, called *neighbourhoods*.
- In a metric space (X, d), the neighbourhood $O_\varepsilon(x)$ of point x can be defined as a set of points a such that

$$d(x, a) < \varepsilon$$

- If X and Y are two topological spaces, then functions $f : X \to Y$ that preserve the topology of X in Y are *continuous* functions.
- Continuous functions map some neighbourhood $O(x) \subseteq X$ into any neighbourhood $O(y) \subseteq Y$:

$$f(x) = y \quad \text{and} \quad f(O(x)) \subseteq O(y)$$

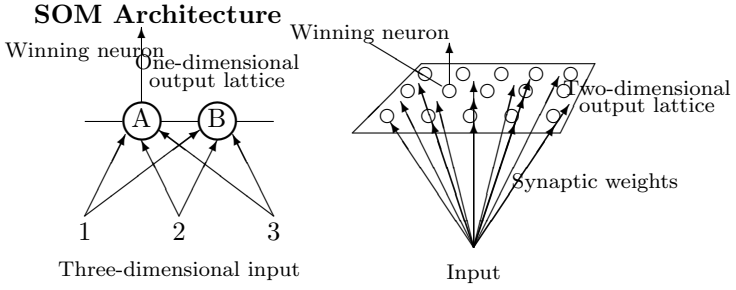
- Thus, to visualise data we need a continuous function from m-dimensional space into a 2-dimensional space:

$$f : \mathbb{R}^m \to \mathbb{R}^2$$

2 Self-Organising Maps

2.1 SOM Architecture

SOM Architecture



- A Self-Organising Map (SOM) is an unsupervised neural network algorithm (Kohonen, 1982) that learns a topology preserving map, and it is used to visualise high-dimensional data.
- It uses a single layer network of artificial neurons (or nodes), which are arranged into an $k \times l$ lattice and are connected to the input space (data).

- The SOM algorithm is designed to establish a correspondence between topologies of the input space (data) and the output lattice (aka Kohonen's topology preserving map).
- Each node has m weights, and together the weights represent some m-dimensional vector: $\mathbf{w}_j = (w_{1j}, \ldots, w_{mj}) \in \mathbb{R}^m$
- The weight vector $\mathbf{w}$ of each neuron can be compared to an m-dimensional vector $\mathbf{x} = (x_1, \ldots, x_m) \in \mathbb{R}^m$ from the input space (data)
- This comparisson is usually done using the Euclidean metric in the input space $\mathbb{R}^m$

$$d_{in}(\mathbf{x}, \mathbf{w}_j) = \sqrt{|x_1 - w_{1j}|^2 + \cdots + |x_m - w_{mj}|^2}$$

- The nodes are arranged into an $k \times l$ lattice (output space).
- The topology in the lattice is defined by another metric, such as the taxi-cab distance:

$$d_{out}(\mathbf{i}, \mathbf{j}) = |i_1 - j_1| + |i_2 - j_2|$$

- The algorithm involves three phases: **competition**, **adaptation** and **co-operation**.

2.2 SOM Algorithm

Competition

- An input vector $\mathbf{x} = (x_1, \ldots, x_m)$ is compared with the weight vector $\mathbf{w}_j = (w_{1j}, \ldots, w_{mj})$ of each node by computing the distance $d(\mathbf{x}, \mathbf{w}_j)$:

$$\begin{aligned} d(\mathbf{x}, \mathbf{w}_1) &= \sqrt{(x_1 - w_{11})^2 + \cdots + (x_m - w_{m1})^2} \\ &\vdots \\ d(\mathbf{x}, \mathbf{w}_n) &= \sqrt{(x_1 - w_{1n})^2 + \cdots + (x_m - w_{mn})^2} \end{aligned}$$

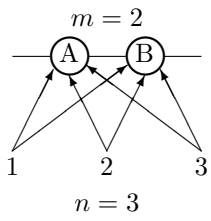
- The *winner* is the node with the weight $\mathbf{w}_j$ closest to the input $\mathbf{x}$ (i.e. shortest $d(\mathbf{x}, \mathbf{w}_j)$).
- Thus, nodes 'compete' in the sense which of the nodes $\mathbf{w}_j$ is more 'similar' to a given input pattern $\mathbf{x}$.

Example 1. Consider SOM with three inputs and two output nodes (A and B). Let

$$\mathbf{w}_A = (2, -1, 3), \quad \mathbf{w}_B = (-2, 0, 1)$$

Find which node is the winner for the input

$$\mathbf{x} = (1, -2, 2)$$



$$\begin{aligned} d(\mathbf{x}, \mathbf{w}_A) &= \sqrt{(1-2)^2 + (-2+1)^2 + (2-3)^2} = \sqrt{3} \\ d(\mathbf{x}, \mathbf{w}_B) &= \sqrt{(1+2)^2 + (-2-0)^2 + (2-1)^2} = \sqrt{14} \end{aligned}$$

- Node A is the winner because it is 'closer' ($\sqrt{3} < \sqrt{14}$)
- What if $\mathbf{x} = (-1, -2, 0)$?

Adaptation

- After the input $\mathbf{x}$ has been presented to SOM, the weights of *all* nodes are *adapted*, so that they become more 'similar' to the input $\mathbf{x}$ vector.
- The adaptation formula for node j is:

$$\mathbf{w}_j^{\text{new}} = \mathbf{w}_j^{\text{old}} + \alpha\, h_{ij} \left[\mathbf{x} - \mathbf{w}_j^{\text{old}}\right] \ ,$$

  where

  - $\mathbf{w}_j$ is the weight vector of node $j \in [1, \ldots, k \times l]$;
  - α is the *learning rate* coefficient;
  - h_{ij} is the *neighbourhood* of node j with respect to the winner i.

Adaptation (cont.)

To understand better the adaptation formula, let us check how the weights change for different values of α and h_{ij}.

$$\mathbf{w}_j^{\text{new}} = \mathbf{w}_j^{\text{old}} + \alpha\, h_{ij} \left[\mathbf{x} - \mathbf{w}_j^{\text{old}}\right] \ ,$$

- Suppose $\alpha = 0$ or $h_{ij} = 0$. Then

$$\mathbf{w}_j^{\text{new}} = \mathbf{w}_j^{\text{old}} + 0 \cdot 0 \left[\mathbf{x} - \mathbf{w}_j^{\text{old}}\right] = \mathbf{w}_j^{\text{old}}$$

  The weight does not change ($\mathbf{w}_j^{\text{new}} = \mathbf{w}_j^{\text{old}}$).

- Suppose $h_{ij} = 1$ and $\alpha = 1$. Then

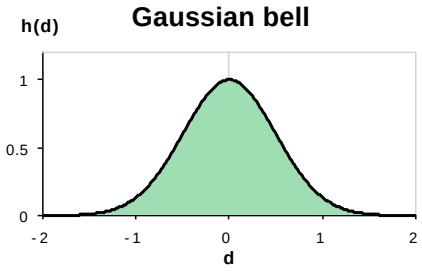
$$\mathbf{w}_j^{\text{new}} = \mathbf{w}_j^{\text{old}} + \mathbf{x} - \mathbf{w}_j^{\text{old}} = \mathbf{x}$$

  The new weight is equal to the input ($\mathbf{w}_j^{\text{new}} = \mathbf{x}$).

Cooperation

- Although weights of *all* nodes are adapted, they do not adapt equally. Adaptation depends on how close the nodes are from the winner in the output lattice.
- If the winner is node i, then the level of adaptation for node j is defined by the *neighbourhood* function $h_{ij} = h(d(i,j))$, where $d(i,j)$ is the distance in the lattice.
- The neighbourhood is defines in such a way that it is smaller as the distance $d(i,j)$ gets larger. For example, the Gaussian bell function

$$h(d(i,j)) = e^{-\frac{d^2(i,j)}{2\sigma^2}}$$



- The winner 'helps' mostly its neighbours to adapt. Note also that the winner is adapted more than any other node (i.e. because $d(i, i) = 0$).

Example 2. Let $\alpha = 0.5$ and $h = 1$, and let us adapt the winning node A from previous example:

$$\mathbf{w}_A = (2, -1, 3)\,, \quad \mathbf{x} = (1, -2, 2)$$

We use adaptation formula: $\mathbf{w}_j^{\text{new}} = \mathbf{w}_j^{\text{old}} + \alpha\, h_{ij}[\mathbf{x} - \mathbf{w}_j^{\text{old}}]$

$$\mathbf{w}_A = \begin{pmatrix} 2 \\ -1 \\ 3 \end{pmatrix} + 0.5 \cdot 1 \cdot \left[\begin{pmatrix} 1 \\ -2 \\ 2 \end{pmatrix} - \begin{pmatrix} 2 \\ -1 \\ 3 \end{pmatrix} \right]$$

$$= \begin{pmatrix} 2 \\ -1 \\ 3 \end{pmatrix} + 0.5 \cdot \begin{pmatrix} -1 \\ -1 \\ -1 \end{pmatrix} = \begin{pmatrix} 1.5 \\ -1.5 \\ 2.5 \end{pmatrix}$$

Training Procedure

- Define the size of the output lattice (i.e. $k \times l$)
- Initialise the weights of the nodes (e.g. randomly)
- Define metrics d_{in}, d_{out} and other parameters (α, h)
- Repeat
  1. Select an input vector $\mathbf{x} \in \mathbb{R}^m$ from data
  2. Find the winning node by minimizing $d_{in}(\mathbf{x}, \mathbf{w}_j)$
  3. Adapt the weights of the winner and its neighbours
- Until the network stabilizes

What Should be the Result?

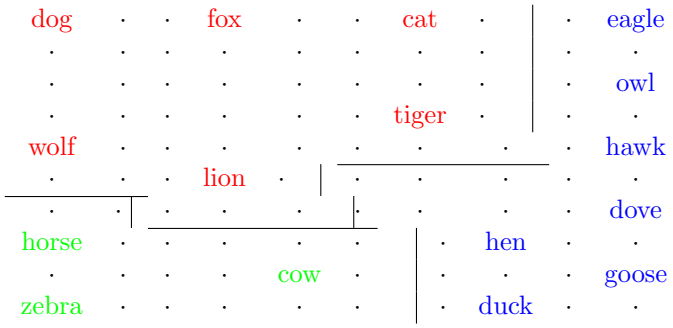
- Initially, there is no relation between the topology (closeness, similarity) in the input space and the topology in the output lattice.
- After training, nodes close to each other in the lattice correspond to points close to each other in the input space.
- The number of input dimensions (m) can be very large (e.g. $m = 100$), so we cannot see the clusters directly.
- The output lattice is usually one or two dimensional, so we can visualise and 'see' the clusters.

3 Applications: Contextual Feature Maps

Example of SOM

	size			legs		hair, hooves, mane, feather				hunt, run, fly, swim			
	s	m	b	2	4								
dove	1	0	0	1	0	0	0	0	1	0	0	1	0
hen	1	0	0	1	0	0	0	0	1	0	0	0	0
duck	1	0	0	1	0	0	0	0	1	0	0	0	1
goose	1	0	0	1	0	0	0	0	1	0	0	1	1
owl	1	0	0	1	0	0	0	0	1	1	0	1	0
hawk	1	0	0	1	0	0	0	0	1	1	0	1	0
eagle	0	1	0	1	0	0	0	0	1	1	0	1	0
fox	0	1	0	0	1	1	0	0	0	1	0	0	0
dog	0	1	0	0	1	1	0	0	0	0	1	0	0
wolf	0	1	0	0	1	1	0	1	0	1	1	0	0
cat	1	0	0	0	1	1	0	0	0	1	0	0	0
tiger	0	0	1	0	1	1	0	0	0	1	1	0	0
lion	0	0	1	0	1	1	0	1	0	1	1	0	0
horse	0	0	1	0	1	1	1	1	0	0	1	0	0
zebra	0	0	1	0	1	1	1	1	0	0	1	0	0
cow	0	0	1	0	1	1	1	0	0	0	0	0	0

Feature Map



Useful Properties of SOM

- Visualisation of multidimensional data.
- Reduces dimensions preserving the topology.
- Useful for clustering.
- Handles missing data
- The learning algorithm is unsupervised.

References

Kohonen, T. (1982). Self-organized formation of topologically correct feature maps. *Biological Cybernetics*, *43*, 59–69.